STATA und R: eine Gegenüberstellung

Mumdzhiev, Milko

Preprint / Preprint
Arbeitspapier / working paper

Nürnberger Beiträge zur Sozial- und Wirtschaftsforschung 01/10

STATA und R
Eine Gegenüberstellung

von Milko Mumdzhiev

Zusammenfassung:
Die Kapitel der Regressionsschätzung, Paneldatenanalyse, der multivariaten Datenanlyse, Survey- und Survivalanalyse, so wie Zeitreihenanlyse werden anhand einer tabellarischen Auflistung von entsprechenden STATA und R-Befehlen vorgestellt. Dabei kann die gewählte Vorgehensweise in STATA zu einem Großteil in R übersetzt und nachvollzogen werden.

Einleitung

Ziel dieser Arbeit war es, STATA 11 Befehlen Entsprechungen aus dem R-Softwarepaket (Version 2.10.1) gegenüberzustellen, und fortgeschrittenen Anfängern in einem dieser Programme eine kleine Übersetzungshilfe in das jeweils andere zu geben. Als Orientierung und Grundlage dienten die offiziellen on- und offline Dokumentationen zu beiden Softwarepaketen, hauptsächlich die STATA Reference Manuals und die R-Packagedokumentationen.

Für die Auswahl der Funktionen war es entscheidend, grob die Lerneinheiten eines ökonometrischen Hauptstudiums beschreiben zu können, somit die Gebiete der Regressionsschätzung, Zeitreihenanalyse, Survivalmodellierung, Panelanalyse, Surveyanalyse und der klassischen multivariaten Datenanalyse nacheinander mehr oder weniger genau abzugehen. Gleichzeitig sollte es möglich sein, die jeweiligen Operationen durch wenige Schritte in beiden Softwareumgebungen nachzuvollziehen; im besten Fall entspricht eine Befehlszeile in STATA einer solchen in R, wobei der Name des Befehls dessen Funktion andeutet. Die Entsprechungen selbst sind oft mit unterschiedlichen Defaultsetzungen implementiert (für R-Befehle wiedergegeben), oder funktionieren in bestimmter Weise nur in der eigenen Softwareumgebung, indem sie einschlägige Operatoren freigeben, lokale Hilfsfunktionen laden usw. Wenige Befehle konnten hinsichtlich aller ihrer Optionen und Spezifikationen gematcht werden.

Die Betrachtung der einzelnen (Sub-)Kommandos bzw. deren Syntax über Hilfedateien und Handbüchern zwecks genauer Optionensetzung, Methodenauswahl, oder zwecks Änderung des Schätzers, der gewünschten Ausgaben etc., sollte hier nicht Platz füllen, wird aber durch eine Verlinkung auf die STATA Web- und Hilfeseiten erleichtert.

In der linken Spalte der Auflistungen wurden die ausgewählten STATA Befehle in alphabetischer Reihenfolge notiert. Die Syntax und ein Beispiel zu einem (zum Teil aus dem Web) noch zu ladenden Datensatz wurden nach STATA Handbuchmuster angefügt. Eine entsprechende R-Spalte enthält die Funktionenbeschreibung, den kompletten (Standard-)Eingabemodus, und ein Beispiel, vom Anschluss der Bibliothek bis zur Schätzung. Oftmals wurden hier weitere Pakete (aus dem Netz) geladen, auf die in der aktuellen Funktion zugegriffen wird. Die Ausgaben in beiden Fällen wurden nicht mit abgebildet, außerdem wurde aus den entsprechenden Prüfgrößen oder Ergebnissen keine Modellgütetestung etc. geliefert.

Ausgelassen wurden Bereiche der klassischen Datenmanipulation, die Speicherung, Einlesen, Variablenberechnung, Fälleaddieren, Rekodieren, Sortieren, Behandlung von missing values, Ausgabenextraktion (in STATA recode, infile, file, generate etc.) abhandeln. Ebenso wurde keine Rücksicht genommen auf Grafikoptionen in Plots, Charts usw.

Informationen über solche Befehle kann man aus den entsprechenden Hilfedateien gewinnen, oder sie in Einführungen zur Software nachlesen. Daneben wurde Matrizenrechnung nicht referiert, sowie Programmiervokabeln und Praktiken, wie if-else, break oder class-Strukturen usw.; alles in allem das, was für Darstellungsweise und Zweck der Arbeit nicht sinnvoll erschien, oder nicht knapp zu benennen war.

Die thematischen Abschnitte, welchen stets ein eigenes Vorwort gewidmet ist, sind im einzelnen:

- Regressionsschätzung, wie sie im STATA Base Reference Manual katalogisiert ist
- Longitudinal / Panel Data Analysis
- Multivariate Datenanalyse
- Befehle aus dem Bereich der Survey / Survival Data Analysis
- Zeitreihenanalyse.

Die Blöcke gehen über einige wenige Seiten, ca. 60 Befehle werden wiedergegeben.
Die einzelnen Einführungen gehen sowohl auf genannte als auch auf nicht genannte Befehle ein, erläutern kurz Optionen zur Schätzung, mögliche *postestimation Schritte* nach dem Befehl, z.B. Modelldiagnosen, und andere Details.
Ein Beispiel soll einführend sein für alle. **Cloglog**-Modelle werden geschätzt. Eckige Klammern in der STATA Syntax werden nicht ausgeschrieben, runde Klammern sehr wohl. **Depvar** steht für die (eine) abhängige Variable, **indepvars** steht für eine oder mehrere unabhängige Variablen. Restriktionensetzung ist möglich, Bestimmung von Gewichten, und andere Spezifikationen (,options) zur Schätzung, zur Ausgabe etc.

cloglog Complementary log-log regression Syntax **cloglog** depvar [indepvars] [if] [in] [weight] [, options] *Fit complementary log-log model* *with robust variance estimates* **sysuse** auto **cloglog** foreign weight mpg, vce(robust)	**cloglog** Complementary log-log regression with glm Syntax **glm(formula, family = binomial, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = glm.control(...), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)** library(HSAUR) glm_1 = glm(ESR ~ fibrinogen, data = plasma, family = binomial(link = "**cloglog**")) summary(glm_1)

1. Regressionsschätzung

Die STATA Befehle alpha, ameans, anova, binreg, bibrobit, bootstrap, clogit, cloglog, doedit, dydx, frontier, glm, ivregress, kdenstiy, ksmirnov, logit, mlogit, mprobit, ologit, oprobit, poisson, probit, qreg, reg3, regress, stem, swilk, summarize, zinb und zip wurden aufgenommen und entsprechenden Befehlen in R gegenübergestellt. Dabei boten sich in STATA und R oft mehrere Alternativen an, eine ganz bestimmte Schätzung durchzuführen, welche dann mehr oder weniger beliebig zitiert wurden (Beispiel: für multinomiale Logitmodelle das R-Package **mlogit** oder das Package **Zelig**). Varianten der Optionensetzung z.B. im **glm**-Befehl (zum Logit-, Poissonmodell etc.) oder *postestimation commands* wurden knappst gehalten, häufig gänzlich unterdrückt; die vorrangige Nutzung einer R-Bibliothek gegenüber einer anderen hatte keine besondere Bedeutung. Die R-Packages car, GMM, sampleSeleciton, sandwich, Design, MASS usw. liefern alternative Anwendungen und Optionen zur Regressionsschätzung, sind aber dennoch unterrepräsentiert, während z.B. das Package Zelig mehrmals angeschlossen wurde.

Explizit nicht aufgenommen und ausgeführt wurden mehrere spezielle Regressionsmodelle oder nicht näher benannte STATA Routinen wie areg, asclogit, asmprobit, eivreg, fracpoly, gmm, heckprob, ivprobit, ivtobit, kwallis, lincom, logistic, mfp, nbreg, nl, nlogit, rocfit, scobit, suest, truncreg, daneben bestimmte Tests oder Graphikbefehle (bitest, diagnostic plots, hausman, heckman, histogramm, lrtest, ttest), *postestimation commands* und andere Schätzer oder Befehle zur Datenhandhabung (z.B. view, ratio, search, ssc, update). *Postestimation commands* zwecks Modelldiagnostik wurden z.T. in der Einleitung exemplarisch gelistet.
Manche Regressionen, z.B. mit Instrumentenvariablen (ivregress in STATA) verwenden 2SLS oder GMM für die Parameterschätzung (wie ivreg im AER-Package in R), bei anderen, z.B. ivprobit stehen die (bedingte) ML-Schätzung oder (Neweys Chi^2-)Two-Step-Estimator zur Auswahl. Weitere implementierte Modelle stellen Verallgemeinerungen anderer dar; bei zu speziellen Ansätzen schien wiederum der Kontext zur Statistiklehre nicht mehr gegeben zu sein. Wenn die Variableneingabe in fertige STATA Routinen sich einfach machte, aber in R dazu einige Rechenschritte mehr erforderlich waren (z.B. das Aufstellen und Optimieren von 2 Gleichungen), dann hat man sich diese der übersichtlichen Kürze und Einfachheit halber gespart, so wie die Nachlieferung bestimmten methodischen Hintergrundes (z.B. zu GMM, Testungen, Tobits, Negbin-Modellen).

Einige Punkte sollen an dieser Stelle ausführlicher referiert werden.

anova

Nach Ausführen von **anova** als der Varianzanalyse (ohne weitere Optionensetzung), mit der Syntax:

anova varname [termlist] [if] [in] [weight] [, options]

One-Way-Anova

webuse systolic

anova systolic drug

aber auch nach anderen vergleichbaren Regressionsschätzungen, kämen modelldiagnostisch folgende *postestimation commands* in Betracht (nur Nennungen):

dfbeta	DFBETA influence statistics
estat hettest	tests for heteroskedasticity
estat imtest	information matrix test
estat ovtest	Ramsey regression specification-error test for omitted variables
estat szroeter	Szroeter's rank test for heteroskedasticity
estat vif	variance inflation factors for the independent variables
acprplot	augmented component-plus-residual plot
avplot	added-variable plot
avplots	all added-variable plots in one image
cprplot	component-plus-residual plot
lvr2plot	leverage-versus-squared-residual plot
rvfplot	residual-versus-fitted plot
rvpplot	residual-versus-predictor plot

Auf Begriff, Syntax und Optionen der *postestimation commands* wird nicht näher eingegangen. Manchmal können diese nicht nach jeder Befehlsoption / Modellspezifikation ausgeführt werden, sei es, weil sie im Spezialfall keinen Sinn machen, oder dass sie so nicht implementiert sind.

Nach einer Schätzung nahezu immer abrufbar sind Informationskriterien wie AIC oder BIC, Wald-Tests, LR-Tests und weitere Modelldiagnosestatistiken.

Typische *postestimation commands* sind (nur Nennungen, ohne Syntax):

estat	AIC, BIC, VCE, and estimation sample summary
hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, inference for linear combinations of coefficients
linktest	link test for model specification
lrtest	likelihood-ratio test
nlcom	point estimates, SE, testing, inference for nonlinear combinations of coefficients
predict	predictions, residuals, influence statistics, and other diagnostic measures
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
suest	seemingly unrelated estimation
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

R bietet vergleichbare Diagnostikbatterien (im Package **lmtest** z.B. LR-Test, Wald-Test, Heteroskedastie-, Linearitätstests etc.), die Hinweise auf die Güte der Modellschätzung liefern können. Ein Paar davon sind tabelliert, wobei in Klammern das jeweilige R-Package genannt ist:

bgtest	Breusch-Godfrey Test (lmtest)
bptest	Breusch-Pagan Test (lmtest)
dfbeta	DBETA (stats)
dwtest	Durbin-Watson Test (lmtest)
plot.lm	a plot of residuals against fitted values, a Normal Q-Q plot and other (stats)
prplot	Partial Residual Plot (faraway)
qq.plot	Quantile-Comparison Plots (car)
lm.influence	This function provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of regression fits (stats)
ls.diag	Computes basic statistics, including standard errors, t- and p-values for the regression coefficients (stats)

logit

Nun wird ein Logitmodell (in STATA mit dem Befehl **logit**) geschätzt, die entsprechende Syntax lautet:

logit depvar [indepvars] [if] [in] [weight] **[, options]**

webuse lbw

logit low age lwt i.race smoke ptl ht ui

Als Optionen (, options) für die Modellschätzung bieten sich an :

noconstant	suppress constant term
offset(varname)	include varname in model with coefficient constrained to 1
asis	retain perfect predictor variables
constraints(constraints)	apply specified linear constraints
collinear	keep collinear variables
vce	vcetype may be robust, bootstrap, jackknife…
level(#)	set confidence level; default is level(95)
or	report odds ratios
nocnsreport	do not display constraints

Logitmodelle könnte man in R beispielsweise über den **zelig**-Befehl (Package Zelig) schätzen.

Die Standardsyntax sieht wie folgt aus:

z.out = **zelig**(Y ~ X1 + X2, model = "**logit**", data = mydata)

Das Setzen von **robust=TRUE** lässt robuste Standardfehler (aus dem Package sandwich) berechnen. Hier stünden mehrere Möglichkeiten zur Auswahl (ohne weitere Erläuterung):

- "vcovHAC"
- "kernHAC"
- "weave"

Konfidenzintervalle würde man grafisch darstellen (verkürzt wiedergegeben) mit:

plot.ci(x, CI = 95, qi = "ev", main = "", ylab = NULL, xlab = NULL, xlim = NULL, ylim = NULL, col = c("red", "blue"), ...)

reg3

Ein letztes Beispiel ist der Befehl **reg3** in STATA, der u.a. über 3SLS ein Gleichungssystem schätzt.

Die Syntax lautet:

reg3 (depvar1 varlist1) (depvar2 varlist2) ...(depvarN varlistN) [if] [in] [weight]

Was den Schätzer angeht, kann man auf mehrere Varianten zugreifen:

3sls	three-stage least squares; the default
2sls	two-stage least squares
ols	ordinary least squares (OLS)
sure	seemingly unrelated regression estimation (SURE)
mvreg	sure with OLS degrees-of-freedom adjustment
corr(correlation)	unstructured or independent correlation structure

Mit dem Befehl **zelig** (im R-Package Zelig) kann man über die **threesls**-Option (Kombination aus 2SLS und SUR) gehen, oder für die *seemingly unrelated regression* die **sur**-Option setzen.

z.out = **zelig**(formula = formula, **model = "threesls"**, data = kmenta)

z.out = **zelig**(formula = formula, **model = "sur",** data = grunfeld)

Allgemein könnte man in R auch mit dem Package **sem** Strukturgleichungsmodelle handhaben.

Nachfolgend setzt der lexikalische Teil dieses Themenbereichs ein. In der linken Spalte stehen die STATA Befehle (welche zur jeweiligen STATA Webseite verlinkt sind), rechts sind die entsprechenden Befehle in R genannt.

alpha Compute inter-item correlations (covariances) and Cronbach's alpha Syntax **alpha** varlist [if] [in] [, options] *Obtain average inter-item covariance and Cronbach's alpha* **webuse** automiss **alpha** price headroom rep78 trunk weight length turn displ	**cronbach** Compute Cronbach's reliability coefficient alpha **Syntax** **cronbach(v1)** library(psy) data(expsy) cronbach(cbind(expsy[,c(1,3:10)],-1*expsy[,2]))
ameans Arithmetic, geometric, and harmonic means Syntax **ameans** [varlist] [if] [in] [weight] [, options] *Compute arithmetic, geometric, and harmonic means* **webuse** systolic **ameans** systolic	**mean** (trimmed) Arithmetic mean **Syntax** **mean(x, ...)** mean(USArrests, trim = 0.2)
anova Analysis of variance and covariance Syntax **anova** varname [termlist] [if] [in] [weight] [, options] *One-Way-Anova* **webuse** systolic **anova** systolic drug *Some postestimation commands* **estat ic** *AIC, BIC* **lvr2plot** *leverage-versus-squared-residual plot* **rvfplot, yline(10)** *residual-versus-fitted plot* **rvpplot drug** *residual-versus-predictor plot*	**anova** Compute an ANOVA-table for linear models **Syntax** **anova(object, ...)** ***where the object contains results returned by a model fitting function (e.g., lm or glm).*** lm1 = lm(Fertility ~ . , data = swiss) summary(lm1) anova(lm1) AIC(lm1) plot.lm(lm1)

binreg Fit generalized linear models for the binomial family Syntax **binreg** depvar [indepvars] [if] [in] [weight] [, options] *Report odds ratios* **webuse** lbw **binreg** low age lwt i.race smoke ptl ht ui, **or**	**glm** Fit generalized linear models **Syntax** **glm(formula, family = gaussian, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = glm.control(...), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)** library(HSAUR) glm_1 = glm(ESR ~ fibrinogen, data = plasma, family = **binomial**()) summary(glm_1)
biprobit Bivariate probit regression Syntax *Bivariate probit model* **biprobit** depvar1 depvar2 [varlist] [if] [in] [weight] [, options] **webuse** school **biprobit** private vote logptax loginc years	**bprobit**-Option (zelig) for bivariate probit regression **Syntax** **z.out = zelig(list(mu1 = Y1 ~ X1 + X2, mu2 = Y2 ~ X1 + X3, rho = ~ 1), model = "bprobit", data = mydata)** library(Zelig) data(sanction) z.out1 = zelig(cbind(import, export) ~ coop + cost + target, **model = "bprobit"**, data = sanction)
bootstrap Bootstrap sampling and estimation Syntax **bootstrap** exp_list [, options eform_option] : **command** *Compute bootstrap estimates* **sysuse** auto **bootstrap**: **regress** mpg weight gear foreign *Bootstrap t statistic using 1000 replications, stratifying on foreign, and saving results in bsauto file* **bootstrap** t=r(t), rep(1000) strata(foreign) saving(bsauto, replace): ttest mpg, by(foreign) unequal	**boot** Generate **R** bootstrap replicates of a statistic applied to data **Syntax** **boot(data, statistic, R, sim="ordinary", stype="i", strata=rep(1,n), L=NULL, m=0, weights=NULL, ran.gen=function(d, p) d, mle=NULL, simple=FALSE, ...)** *usual bootstrap of the ratio of means using the city data* library(boot) ratio = function(d, w) sum(d$x * w)/sum(d$u * w) boot(city, ratio, R=999, stype="w")

clogit Conditional (fixed-effects) logistic regression Syntax **clogit** depvar [indepvars] [if] [in] [weight] , group(varname) [options] *Fit conditional logistic regression (panel data)* **webuse** union, clear **clogit** union age grade not_smsa, group(idcode)	**clogit** Estimate a logistic regression model by maximising the conditional likelihood **Syntax** **library(survival)** **clogit(formula, data, method=c("exact", "approximate"),** **na.action=getOption("na.action"),** **subset=NULL,control=coxph.control())**
cloglog Complementary log-log regression Syntax **cloglog** depvar [indepvars] [if] [in] [weight] [, options] *Fit complementary log-log model with robust variance estimates* **sysuse** auto **cloglog** foreign weight mpg, vce(robust)	**cloglog** Complementary log-log regression with glm Syntax **glm(formula, family = binomial, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = glm.control(...), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)** library(HSAUR) glm_1 = glm(ESR ~ fibrinogen, data = plasma, family = binomial(link = "**cloglog**")) summary(glm_1)
doedit Edit do-files and other text files Syntax **doedit** [filename]	**edit** Invoke a text editor **Syntax** **edit(name = NULL, file = "", title = NULL, editor = getOption("editor"), ...)** edit(glm) func=edit(glm)
dydx Calculate numeric derivatives Syntax **dydx** yvar xvar [if] [in] , generate(newvar) [dydx_options] *1. create 100 obs on x, 0 to 4*pi* *2. generate y = f(x)* *3. create derivative of function* **range** x 0 12.56 100 **generate** y = exp(-x/6)*sin(x) **dydx** y x, gen(yprime)	**grad** Calculate a numerical approximation of the first derivative of func at the point x **Syntax** **grad(func, x, method="Richardson", method.args =list(), ...)** library(numDeriv) grad(sin, pi)

frontier Stochastic frontier models Syntax **frontier** depvar [indepvars] [if] [in] [weight] [, options] *Cobb-Douglas production function with exponential distribution for inefficiency term* **webuse** greene9 **frontier** lnv lnk lnl, dist(exponential)	**sfa** MLE of stochastic frontier and cost models **Syntax** **sfa(formula, data = sys.frame(sys.parent()), ineffDecrease = TRUE, truncNorm = FALSE, timeEffect = FALSE, startVal = NULL, tol = 0.00001, maxit = 1000, muBound = 2, bignum = 1.0E+16, searchStep = 0.00001, searchTol = 0.001, searchScale = NA, gridSize = 0.1, gridDouble = TRUE, printIter = 0)** *Cobb-Douglas production frontier* library(frontier) data(front41Data) cobbDouglas = sfa(log(output) ~ log(capital) + log(labour), data = front41Data) summary(cobbDouglas)
glm Generalized linear models Syntax **glm** depvar [indepvars] [if] [in] [weight] [, options] *Generalized linear model with binomial family and cloglog link* **webuse** ldose **glm** r ldose, family(binomial n) link(cloglog)	**glm** Generalized linear models **Syntax** **glm(formula, family = gaussian, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = glm.control(...), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)**
ivregress Single-equation instrumental variables regression Syntax **ivregress** estimator depvar [varlist1] (varlist2 = varlist_iv) [if] [in] [weight] [, options] *Fit a regression via 2SLS, requesting small sample statistics* **webuse** hsng2 **ivregress** 2sls rent pcturban (hsngval = faminc i.region), small	**twosls**-Option (zelig) for consistent estimates for linear regression models with some explanatory variable correlated with the error term using instrumental variables **Syntax** **zelig(formula = fml, model = "twosls", data = mydata)** library(Zelig) data(klein) formula = list(mu1 = C ~ Wtot + P + P1, mu2 = I ~ P + P1 + K1, mu3 = Wp ~ X + X1 + Tm, inst = ~P1 + K1 + X1 + Tm + Wg + G) z.out = zelig(formula = formula, **model = "twosls"**, data = klein) summary(z.out)

kdensity Univariate kernel density estimation Syntax **kdensity** varname [if] [in] [weight] [, options] **sysuse** auto **kdensity** length	**kde** Perform KDE for 1-6 dimensional data **Syntax** **kde(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points, binned=FALSE, bgridsize, positive=FALSE, adj.positive, w)** *Univariate example* library(ks) x = rnorm.mixt(n=100, mus=1, sigmas=1, props=1) fhat = kde(x=x, h=hpi(x)) plot(fhat)
ksmirnov Kolmogorov–Smirnov equality of distributions test Syntax *One-sample Kolmogorov-Smirnov test* **ksmirnov** varname = exp [if] [in] *One-sample KS test* **webuse** ksxmpl **summarize** x **ksmirnov** x = normal((x-r(mean))/r(sd)) *Two-sample KS test* **ksmirnov** x, by(group)	**ks.test** One or two sample KS-tests **Syntax** **ks.test(x, y, ...,** **alternative = c("two.sided", "less", "greater"),** **exact = NULL)** *Do x and y come from the same distribution?* ks.test(x, y)
logit Logistic regression for binary outcomes, reporting coefficients Syntax **logit** depvar [indepvars] [if] [in] [weight] [, options] **webuse** lbw **logit** low age lwt i.race smoke ptl ht ui	**logit**-Option (zelig) for logistic regression **Syntax** **z.out = zelig(Y ~ X1 + X2, model = "logit",** **data = mydata)** library(Zelig) data(turnout) z.out1 = zelig(vote ~ age + race, **model = "logit"**, data = turnout) summary(z.out1)
mlogit Multinomial logistic regression (categorical outcomes) Syntax **mlogit** depvar [indepvars] [if] [in] [weight] [, options] **webuse** sysdsn1 **mlogit** insure age male nonwhite i.site	**mlogit**-Option (zelig) uses the multinomial logit distribution to model unordered categorical variables **Syntax** **z.out = zelig(as.factor(Y) ~ X1 + X2,** **model = "mlogit", data = mydata)** library(Zelig) data(mexico) z.out1 = zelig(as.factor(vote88) ~ pristr + othcok + othsocok, **model = "mlogit"**, data = mexico) summary(z.out1)

mprobit Multinomial probit regression (categorical outcomes) Syntax **mprobit** depvar [indepvars] [if] [in] [weight] [, options] **webuse** sysdsn1 **mprobit** insure age male nonwhite i.site	**mnp** (Bayesian) Multinomial probit regression **Syntax** **mnp(formula, data = parent.frame(), choiceX = NULL, cXnames = NULL, base = NULL, latent = FALSE, invcdf = FALSE, trace = TRUE, n.draws = 5000, p.var = "Inf", p.df = n.dim+1, p.scale = 1, coef.start = 0, cov.start = 1, burnin = 0, thin = 0, verbose = FALSE)** *run the multinomial probit model with ordered preferences* library(MNP) data(japan) res2 = mnp(cbind(LDP, NFP, SKG, JCP) ~ gender + education + age, data = japan, verbose = TRUE)
ologit Ordered logistic regression Syntax **ologit** depvar [indepvars] [if] [in] [weight] [, options] **webuse** fullauto **ologit** rep77 foreign length mpg	**ologit**-Option (zelig) for ordinal logit regression model if dependent variable is ordered and categorical **Syntax** **z.out = zelig(as.factor(Y) ~ X1 + X2, model = "ologit", data = mydata)** library(Zelig) data(sanction) sanction$ncost = factor(sanction$ncost, ordered = TRUE, levels = c("net gain", "little effect", "modest loss", "major loss")) z.out = zelig(ncost ~ mil + coop, **model = "ologit"**, data = sanction)
oprobit Ordered probit regression Syntax **oprobit** depvar [indepvars] [if] [in] [weight] [, options] **webuse** fullauto **oprobit** rep77 foreign length mpg	**oprobit**-Option (zelig): Ordinal probit regression for ordered categorical dependent variables **Syntax** **z.out = zelig(as.factor(Y) ~ X1 + X2, model = "oprobit", data = mydata)** library(Zelig) data(sanction) sanction$ncost = factor(sanction$ncost, ordered = TRUE, levels = c("net gain", "little effect", "modest loss", "major loss")) z.out = zelig(ncost ~ mil + coop, **model = "oprobit"**, data = sanction) summary(z.out)

poisson Poisson regression (count outcomes) Syntax **poisson** depvar [indepvars] [if] [in] [weight] [, options] **webuse** dollhill3 **poisson** deaths smokes i.agecat, exposure(pyears)	**poisson**-Option (zelig) for poisson regression **Syntax** **z.out = zelig(Y ~ X1 + X2, model = "poisson", data = mydata)** library(Zelig) data(sanction) z.out1 = zelig(num ~ target + coop, model = "poisson", data = sanction) summary(z.out1)
probit Probit regression for binary outcomes Syntax **probit** depvar [indepvars] [if] [in] [weight] [, options] **sysuse** auto **probit** foreign weight mpg	**probit**-Option (zelig) for probit regression **Syntax** **z.out = zelig(Y ~ X1 + X2, model = "probit", data = mydata)** library(Zelig) data(turnout) z.out2= zelig(vote ~ race + educate, **model = "probit"**, data = turnout)
qreg Quantile regression models Syntax **qreg** depvar [indepvars] [if] [in] [weight] [, qreg_options] *Median regression* **sysuse** auto **qreg** price weight length foreign	**quantile**-Option (zelig) for quantile regression models **Syntax** **z.out = zelig(Y ~ X1 + X2, model = "quantile", data = mydata, tau = 0.5)** library(Zelig) data(stackloss) z.out11 = zelig(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., **model = "quantile"**, data = stackloss, tau = 0.5)
reg3 Three-stage estimation for systems of simultaneous equations Basic syntax **reg3** (depvar1 varlist1) (depvar2 varlist2) ...(depvarN varlistN) [if] [in] [weight] **webuse** klein **reg3** (consump wagepriv wagegovt) (wagepriv consump govt capital1)	**threesls**-Option (zelig) is a combination of two stage least squares and seemingly unrelated regression **Syntax** **fml = list(mu1 = Y1 ~ X1 + Z1, mu2 = Y2 ~ X2 + Z2, inst1 = Z1 ~ W1 + X1, inst2 = Z2 ~ W2 + X2)** **z.out = zelig(formula = fml, model = "threesls", data = mydata)** library(Zelig) data(kmenta) formula = list(mu1 = q ~ p + d, mu2 = q ~ p + f + a, inst = ~d + f + a) z.out = zelig(formula = formula, **model = "threesls"**, data = kmenta) summary(z.out)

regress Linear regression Syntax **regress** depvar [indepvars] [if] [in] [weight] [, options] *Suppress intercept term* **sysuse** auto **regress** weight length, noconstant	**lm** Fit linear models **Syntax** **lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)** data(swiss) summary(lm(Fertility ~ . , data = swiss))
stem Stem-and-leaf displays Syntax **stem** varname [if] [in] [, options] **webuse** stemxmpl **stem** x	**stem** Stem-and-leaf plot of the values in x **Syntax** **stem(x, scale = 1, width = 80, atom = 1e-08)** stem(islands)
swilk Shapiro–Wilk tests for normality Syntax *Shapiro-Wilk normality test* **swilk** varlist [if] [in] [, options] **sysuse** auto **swilk** mpg	**shapiro.test** Perform the Shapiro-Wilk test of normality Syntax **shapiro.test(x)**
summarize Summary statistics Syntax **summarize** [varlist] [if] [in] [weight] [, options] **sysuse** auto **summarize** **summarize** mpg weight if foreign, detail	**summary** Summary statistics of object **Syntax** **summary(object, ..., digits = max(3, getOption("digits")-3))**
zinb Zero-inflated negative binomial regression Syntax **zinb** depvar [indepvars] [if] [in] [weight], inflate(varlist[, offset(varname)]|_cons) [options] **webuse** fish **zinb** count persons livebait, inflate(child camper)	**zeroinfl** Fit zero-inflated regression models for count data via maximum likelihood **Syntax** **zeroinfl(formula, data, subset, na.action, weights, offset,** **dist = c("poisson", "negbin", "geometric"),** **link = c("logit", "probit", "cloglog", "cauchit", "log"), control = zeroinfl.control(...),** **model = TRUE, y = TRUE, x = FALSE, ...)** *with simple inflation (no regressors for zero component)* library (pscl) data(bioChemists) fm_zip = zeroinfl(art ~ fem + mar + kid5 + phd + ment | 1, data = bioChemists, **dist = "negbin"**)
zip Zero-inflated poisson regression Syntax **zip** depvar [indepvars] [if] [in] [weight], inflate(varlist[, offset(varname)]|_cons) [options] **webuse** fish **zip** count persons livebait, inflate(child camper)	

2. Panelanalyse

Aufgenommen wurden in diesem Abschnitt die Befehle xtfrontier, xtgls, xthtaylor, xtivreg, xtset.
Bevor man zur Analyse selbst übergeht, definiert man über:
xtset panelvar timevar [, tsoptions]
die vorliegenden Daten als Paneldaten, und kann danach Befehle der Form xtreg, xtline, xtlogit anwenden, bestimmte Operatoren werden aktiviert, jeweilige *postestimation commands* zur Verfügung gestellt usw.
Das Package **plm** wurde zum Großteil verwendet, um STATA Implementierungen in R wiederzugeben. **Pdata.frame** oder **plm.data** "formatieren" in R den jeweiligen Datensatz in ein Objekt mit einer ID- und einer Zeitvariable, welches mit entsprechenden Befehlen zur Regressionsschätzung, z.B. **plm,** angegangen werden kann.
Zur Modellschätzung in STATA bieten sich xtabond, xtcloglog, xtfrontier, xtgee, xtgls, xthtaylor, xtintreg, xtivreg, xtlogit, xtmixed, xtnbreg, xtprobit, xtreg und weitere an, wobei die **xt**-Vorsilbe auf Panelmodellierung nach Ausführen von **xtset** hinweist. Auch hier sind Übereinstimmungen von STATA und R-Befehlen nicht notwendigerweise über alle Modellunteroptionen gegeben.

Beispielhaft sollen einige Befehle auskommentiert werden.

xtreg

STATA bietet **xtreg** für Modellschätzung mit fixed-, between-effects und anderen Effekten.
Die Syntax für das **RE**(random-effects)-Modell lautet:
xtreg depvar [indepvars] [if] [in] **[, re RE_options]**
Mögliche Optionen sind für diesen Fall:

re	use random-effects estimator; the default
sa	use Swamy-Arora estimator of the variance components
vce(vcetype)	vcetype may be conventional, robust, jackknife…
level(#)	set confidence level; default is level(95)
theta	report theta
display_options	control spacing and display of omitted variables and base and empty cell

Nach der (RE-)Modellschätzung könnte man z.B. anwenden:

xttest0	Breusch and Pagan LM test for random effects
estat	AIC, BIC, VCE, and estimation sample summary
estimates	cataloging estimation results
hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects
nlcom	point estimates, SE, testing, and inference for nonlinear combinations of coefficients
predict	predictions, residuals, influence statistics, and other diagnostic measures
predictnl	point estimates, SE, testing, and inference for generalized predictions
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

xtgls

Der Befehl **xtgls** schätzt lineare Modelle mit dem FGLS-Schätzer. Die Syntax lautet:

xtgls depvar [indepvars] [if] [in] [weight] **[, options]**

Fit panel-data model with heteroskedasticity across panels

webuse invest2

xtset company time

xtgls invest market stock, **panels(hetero)**

Hier wird ein Spezialfall geschätzt, dabei ist die Option **, panels**(**hetero)** die Abkürzung von **panels(heteroskedastic)**. Weitere Optionen zum **xtgls**-Befehl sind weiter unten tabelliert.

Optionen zu **xtgls** sind u.a.:

noconstant	suppress constant term
panels(iid)	use i.i.d. error structure
panels(heteroskedastic)	use heteroskedastic but uncorrelated error structure
panels(correlated)	use heteroskedastic and correlated error structure
corr(independent)	use independent autocorrelation structure
corr(psar1)	use panel-specific AR(1) autocorrelation structure
rhotype(calc)	specify method to compute autocorrelation parameter
igls	use iterated GLS estimator instead of two-step GLS estimator
level(#)	level(#) set confidence level; default is level(95)

Als *postestimation commands* bieten sich an:

estat	AIC, BIC, VCE, and estimation sample summary
estimates	cataloging estimation results
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects
nlcom	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
predict	predictions, residuals, influence statistics, and other diagnostic measures
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

xtivreg

Mit **xtivreg** kann man in STATA Modelle mit endogenen / instrumentalisierten Variablen schätzen; hier ist die Syntax für ein Modell mit random-effects:

xtivreg depvar [varlist_1] (varlist_2 = varlist_iv) [if] [in] **[, re RE_options]**

Als Modelloptionen bieten sich u.a. an:

re	use random-effects estimator; the default
ec2sls	use Baltagi's EC2SLS random-effects estimator
nosa	use the Baltagi-Chang estimators of the variance components
regress	treat covariates as exogenous and ignore instrumental variables

Um Paneldaten in R zu modellieren, lädt man das **plm**-Package.

pggls verwendet den FGLS-Schätzer; **plm** kann sowohl die üblichen Effekte, als auch Instrumentenvariablen einbeziehen, und bietet für den RE-Fall 4 Schätzer, u.a. Swamy und Aroras Schätzer. Hausman und Taylors Schätzer kann ebenfalls gefordert werden.

Beispiele für eine FGLS-Schätzung und eine Regression mit **plm** in R werden nachfolgend gegeben.

pggls-Syntax

pggls(formula, data, subset, na.action, effect = c("individual","time"), model = c("within","random"), index = NULL, ...)

```
library(plm)
data("Produc", package = "plm")
zz = pggls(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model = "random")
```

plm-Syntax

plm(**formula**, data, subset, na.action, effect = c("individual","time","twoways"), model = c**("within","random","ht","between", "pooling","fd")**,

random.method = c**("swar","walhus","amemiya","nerlove"),** inst.method = c("bvk","baltagi"), index = NULL, ...)

formula *with IV: y~x1+x2+x3 | x3+z1+z2, with z-variables being external, x1 and x2 endogenous*

Beispiel:

```
library(plm)
data(Grunfeld)

grun.fe = plm(inv ~ value + capital, data =Grunfeld, model = "within")
```

Verfügbar wären zwecks Modelldiagnose die Testverfahren:

pFtest	a simple test for the presence of individual (or/and time) effects based on the comparison of the pooling and the within models
plmtest	a set of likelihood ratio tests for the presence of individual (or/and time) effects based on the comparison of the random and the pooling model
phausman	a Hausman-test for the correlation between explanatory variables and individual (or/and time) effects, based on the comparison of the within and the random models.

Visualisierungsbefehle, Tabellierungstools, Diagnosetestungen, die Handhabung bestimmter Ergebnisse aus den Ausgaben usw. wurden für STATA und R nicht extra erläutert.
Ebenso ausgelassen wurden die Befehle **xtgee, xtlogit, xtmixed, xtprobit, xtpoisson, xttobit**, welche Logit-, Mixed-effects-, Probitmodelle u.a. für Paneldaten und Variablen schätzen.
Vergleichbare Routinen / Optionen aus den R-Packages **gee** (Genearal Equation Estimation) oder **Zelig**, wie **logit.gee** oder **logit.mixed**, **poisson.gee** oder **poisson.mixed** würde man in folgender Schreibweise verwenden:

```
library(Zelig)
z.out = zelig(Y ~ X1 + X2, model = "logit.gee", id = "X3", data = mydata)
```

Das Package **lme4** (oder **nlme**) bietet weitere Routinen, um LM oder GLM mit mixed-effects zu schätzen.
Beispiel:

```
library(lme4)
gm1 = glmer(cbind(incidence, size - incidence) ~ period + (1 | herd), family = binomial, data = cbpp))
```

Es folgt die parallele Auflistung von STATA und R-Befehlen.

xtfrontier Stochastic frontier models for panel data Syntax *Time-invariant model* **xtfrontier** depvar [indepvars] [if] [in] [weight] , **ti** [ti_options] *Time-invariant model* **webuse** xtfrontier1 **xtfrontier** lnwidgets lnmachines lnworkers, **ti**	**sfa** Maximum likelihood estimation of stochastic frontier production and cost functions **Syntax** **sfa(formula, data = sys.frame(sys.parent()), ineffDecrease = TRUE, truncNorm = FALSE, *timeEffect* = *FALSE*, startVal = NULL, tol = 0.00001, maxit = 1000, muBound = 2, bignum = 1.0E+16, searchStep = 0.00001, searchTol = 0.001, searchScale = NA, gridSize = 0.1, gridDouble = TRUE, printIter = 0)** library(frontier) data(riceProdPhil) riceProdPhil = plm.data(riceProdPhil, c("FMERCODE", "YEARDUM")) rice = sfa(log(PROD) ~ log(AREA) + log(LABOR) + log(NPK), data = riceProdPhil)
xtgls Fit panel-data models by using GLS Syntax **xtgls** depvar [indepvars] [if] [in] [weight] [, options] *Fit panel-data model with heteroskedasticity across panels* **webuse** invest2 **xtset** company time **xtgls** invest market stock, panels(hetero)	**pggls** General (Feasible-)GLS estimators for panel data (balanced or unbalanced) **Syntax** **pggls(formula, data, subset, na.action, effect = c("individual","time"), model = c("within","random"), index = NULL, ...)** library(plm) data("Produc", package = "plm") zz = pggls(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model = "random") summary(zz)
xthtaylor Hausman–Taylor estimator for error components models Syntax **xthtaylor** depvar indepvars [if] [in] [weight] , endog(varlist) [options] **webuse** psidextract **xthtaylor** lwage wks south smsa ms exp exp2 occ ind union fem blk ed, endog(exp exp2 occ ind union ed) constant(fem blk ed)	**pht** Hausman-Taylor estimator as instrumental variable estimator without external instruments **Syntax** **pht(formula, data, subset, na.action, index = NULL, ...)** library(plm) data("Wages", package = "plm") ht = plm(lwage~wks+south+smsa+ married+exp+I(exp^2)+ bluecol+ind+ union+sex+black +ed | sex+black+bluecol+south+smsa+ind, data=Wages, **model="ht"**, index=595) summary(ht)

xtivreg Instrumental variables and two-stage least squares for panel-data models Syntax *Fixed-effects (FE) model* **xtivreg** depvar [varlist_1] (varlist_2 = varlist_iv) [if] [in] , fe [FE_options] *Fixed-effects model* **webuse** nlswork **xtivreg** ln_w age c.age#c.age not_smsa (tenure = union south), fe	**plm** Linear models for panel data estimated using the **lm** function on transformed data - the fixed effects model (**within**) - the pooling model (**pooling**) - the first-difference model (**fd**) - the between model (**between**) - the error components model (**random**) **Syntax** **plm(formula, data, subset, na.action, effect = c("individual","time","twoways"), model = c("within","random","ht","between", "pooling","fd"), random.method = c("swar","walhus","amemiya", "nerlove"), inst.method = c("bvk","baltagi"), index = NULL, ...)** *formula with IV: y~x1+x2+x3 | x3+z1+z2, with z-variables being external instruments, x1 and x2 endogenous* library(plm) data(Grunfeld) grun.fe = plm(inv ~ value + capital, data = Grunfeld, model = "**within**")
xtset Declare data to be panel data Syntax *Declare panel and time variable* **xtset** panelvar timevar [, tsoptions] **webuse** invest2 **xtset** company time	**pdata.frame** An object of this class is a data.frame with an attribute that describes its time and individual dimensions **Syntax** **pdata.frame(x, index = NULL, drop.index = FALSE, row.names = TRUE)** *Gasoline contains two variables which are individual and time indexes* library(plm) data("Gasoline", package = "plm") Gas = pdata.frame(Gasoline, c("country","year"), drop = TRUE)

3. Multivariate Datenanalyse

In diesem Abschnitt aufgenommen wurden die Befehle ca, candisc, canon, cluster kmeans, cluster singlelinkage, cluster medianlinkage und weitere hierarch. Clusteranalysen, daneben factor, mca, mds und pca. Auf jeweilige *postesimation commands*, grafische Optionen oder Datenmanipulationen wurde nicht näher eingegangen. Ausgelassen wurden u.a. die STATA Befehle hotteling, manova, procrustes, scoreplot.
Die R-Packages MASS, class, SensoMineR, ICSNP, FactoMineR u.a. liefern alternative Befehle und Versionen zur multivariaten Datenanalyse, die an dieser Stelle jedoch nur zu einem geringen Teil ausgeschöpft wurden.
Als Argument können einschlägige STATA Befehle Daten, im Sinne von Variablenangaben, oder eine Matrix fordern. Die Variableneingabe wurde stets vorgezogen.

Die Korrespondenzanalyse **ca** wird aufgerufen über die Syntax:

ca rowvar colvar [if] [in] [weight] [, options]
Um **camat** auszuführen, muss eine Matrix **matname** eingegeben werden:
camat matname [, options]

Hervorgehoben werden einige Datenanalysemethoden.

cluster averagelinkage

Die average-linkage-Clusteranalyse hat die Syntax:
cluster averagelinkage [varlist] [if] [in] [, cluster_options]
Folgende Optionen wären u.a. möglich:

measure(bspw. L2)	similarity or dissimilarity measure
name(bspw.L2slnk)	name of resulting cluster analysis

Mit **name** vergibt man einen Namen, und lässt zur Clusteranalyse ein Dendrogramm zeichnen:
webuse labtech
cluster averagelinkage x1 x2 x3 x4, **name**(L2slnk)
cluster list L2clnk
cluster dendrogram L2slnk, xlabel(, angle(90) labsize(*.75))

In R kann über den Befehl **hclust** gleiches erreicht werden.

hc = **hclust**(dist(USArrests), "**ave**")

plot(hc)

plot(hc, hang = -1)

factor

Eine Faktorenanalyse kann man in STATA über die Syntax aufrufen:

factor varlist [if] [in] [weight] [, method options]

Optionen für die Analyse können sein:

pf	principal factor; the default
pcf	principal-component factor
ipf	iterated principal factor
ml	maximum-likelihood factor
factors(#)	maximum number of factors to be retained
mineigen(#)	minimum value of eigenvalues to be retained
citerate(#)	communality reestimation iterations (ipf only)
altdivisor	use trace of correlation matrix as the divisor for reported proportions
random	use random starting values (ml only); seldom used
seed(seed)	random-number seed (ml with protect() or random only)
maximize_options	control the maximization process; seldom used (ml only)

Als *postestimation commands* kommen in Frage:

estat anti	anti-image correlation and covariance matrices
estat common	correlation matrix of the common factors
estat factors	AIC and BIC model-selection criteria for different numbers of factors
estat kmo	Kaiser-Meyer-Olkin measure of sampling adequacy
estat residuals	matrix of correlation residuals
estat rotatecompare	compare rotated and unrotated loadings
estat smc	squared multiple correlations between each variable and the rest
estat structure	correlations between variables and common factors
loadingplot	plot factor loadings
rotate	rotate factor loadings
screeplot	plot eigenvalues

Der entsprechende R-Befehl für eine Faktorenanalyse lautet:

factanal (x, factors, data = NULL, covmat = NULL, n.obs = NA, subset, na.action,

start = NULL, scores = c("none","regression", "Bartlett"), rotation = "varimax", control = NULL,...)

Nachfolgend werden ausgewählte STATA und R-Befehle zur multivariaten Datenanalyse gelistet.

ca Simple correspondence analysis Syntax *Simple correspondence analysis of two categorical variables* **ca** rowvar colvar [if] [in] [weight] [, options] **webuse** ca_smoking **ca** rank smoking	**anacor** Simple and canonical CA **Syntax** **anacor(tab, ndim = 2, row.covariates, col.covariates, scaling = c("Benzecri","Benzecri"), eps = 1e-06)** library(anacor) data(tocher) res = anacor(tocher, scaling = c("standard", "centroid")) res
candisc Canonical linear discriminant analysis Syntax **candisc** varlist [if] [in] [weight], group(groupvar) [options] **webuse** rootstock **candisc** y1 y2 y3 y4, group(rootstock)	**candisc** Perform a generalized canonical discriminant analysis for one term in a multivariate linear model **Syntax** **candisc(mod, term, type = "2", manova, ndim = rank, ...)** library(candisc) iris.mod = lm(cbind(Petal.Length, Sepal.Length, Petal.Width, Sepal.Width) ~ Species, data=iris) iris.can = candisc(iris.mod, data=iris)
canon Canonical correlations Syntax **canon** (varlist1) (varlist2) [if] [in] [weight] [, options] **sysuse** auto **canon** (length weight headroom trunk) (displ mpg gear_ratio turn)	**cancor** Perform canonical correlations analysis **Syntax** **cancor(x, y, xcenter = TRUE, ycenter = TRUE)** pop = LifeCycleSavings[, 2:3] oec = LifeCycleSavings[, -(2:3)] cancor(pop, oec)
cluster kmeans Kmeans cluster analysis Syntax **cluster kmeans** [varlist] [if] [in] , k(#) [options] *CA creating 8 groups* **webuse** labtech **cluster kmeans** x1 x2 x3 x4, k(8)	**kmeans** Perform k-means clustering on a data matrix **Syntax** **kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))** x = rbind(matrix(rnorm(100, sd = 0.3), ncol = 2), matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2)) colnames(x) = c("x", "y") cl = kmeans(x, 2) plot(x, col = cl$cluster) points(cl$centers, col = 1:2, pch = 8, cex=2)

cluster linkage Hierarchical cluster analysis (CA) Syntax **cluster *linkage*** [varlist] [if] [in] [, cluster_options] *where **linkage** stands for* **singlelinkage** **averagelinkage** **completelinkage** **waveragelinkage** **medianlinkage** **centroidlinkage** *Single linkage CA with cluster trees drawn* **webuse** labtech **cluster singlelinkage** x1 x2 x3 x4, name(L2slnk) **cluster dendrogram** L2slnk, xlabel(, angle(90) labsize(*.75))	**hclust** Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it **Syntax** **hclust(d, method = "complete", members=NULL)** ***method =the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid".*** hc = hclust(dist(USArrests), "ave") plot(hc) plot(hc, hang = -1)
factor Factor analysis Syntax **factor** varlist [if] [in] [weight] [, method options] *Maximum-likelihood factors, keep 2* **webuse** bg2 **factor** bg2cost1-bg2cost6, factors(2) ml	**factanal** Perform maximum-likelihood factor analysis on a covariance matrix or data matrix **Syntax** **factanal(x, factors, data = NULL, covmat = NULL, n.obs = NA, subset, na.action, start = NULL, scores = c("none", "regression", "Bartlett"), rotation = "varimax", control = NULL, ...)** v1 = c(1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,4,5,6) v2 = c(1,2,1,1,1,1,2,1,2,1,3,4,3,3,3,4,6,5) v3 = c(3,3,3,3,3,1,1,1,1,1,1,1,1,1,1,5,4,6) v4 = c(3,3,4,3,3,1,1,2,1,1,1,1,2,1,1,5,6,4) v5 = c(1,1,1,1,1,3,3,3,3,3,1,1,1,1,1,6,4,5) v6 = c(1,1,1,2,1,3,3,3,4,3,1,1,1,2,1,6,5,4) m1 = cbind(v1,v2,v3,v4,v5,v6) cor(m1) factanal(m1, factors=3)
mca Multiple and joint correspondence analysis Syntax *For two or more categorical variable*s **mca** varlist [if] [in] [weight][, options] *Analyzing the indicator matrix of the data, extracting three dimensions, compact output* **webuse** issp93a **mca** A B C D, method(indicator) dim(3) compact	**mjca** Computation of multiple and joint correspondence analysis **Syntax** **mjca(obj, nd = 2, lambda = "adjusted", supcol = NA, subsetcol = NA, ps = "", maxit = 50, epsilon = 0.0001)** *Joint correspondence analysis*: library(ca) library(MASS) data(farms) mjca(farms, lambda = "JCA")

mds Multidimensional scaling for two-way data Syntax **mds** varlist [if] [in] , id(varname) [options] **sysuse** auto **mds** price-gear, id(make)	**cmdscale** Classical multidimensional scaling of a data matrix, also known as principal coordinates analysis **Syntax** **cmdscale(d, k = 2, eig = FALSE, add = FALSE, x.ret = FALSE)** cmdsE = cmdscale(eurodist, k=20, add = TRUE, eig = TRUE, x.ret = TRUE)
pca Principal component analysis (of data) Syntax **pca** varlist [if] [in] [weight] [, options] *Assuming multivariate normality of data* **webuse** bg2 **pca** bg2cost*, vce(normal)	**prcomp** Perform a principal components analysis on the given data matrix **Syntax** **prcomp(formula, data = NULL, subset, na.action, ...)** prcomp(~ Murder + Assault + Rape, data = USArrests, scale = TRUE) plot(prcomp(USArrests))

4. Surveyanalyse

Aufgenommen wurden separat der STATA Befehl **svyset**, welcher das Surveydesign am vorliegenden Datensatz definiert, sowie listenweise Befehle zur Modellschätzung in der Schreibart **svy: command,** z.B. **svy: regress** oder **svy: poisson.**

In R wurde parallel das Package **survey** referiert.

Wie üblich wurden *postestimation commands*, etwaige Graphikoptionen oder Testverfahren ausgelassen.

Einige Beispiele sollen in diesen Abschnitt einleiten.

svyset

svyset bestimmt das Surveydesign, setzt ID-Variablen (primary sampling units), den Varianzschätzer, aktiviert Befehle der Form **svy: logit** etc.

Single-stage syntax

svyset [psu] [weight] [, design_options options]

One-stage clustered design with stratification

webuse stage5a

svyset su1 [pweight=pw], strata(strata)

Syntaxoptionen wären:

strata(varname)	variable identifying strata
fpc(varname)	finite population correction
brrweight(varlist)	balanced repeated replicate (BRR) weights
fay(#)	Fay's adjustment
jkrweight(varlist, ...)	jackknife replicate weights
vce(linearized)	Taylor linearized variance estimation
vce(brr)	balanced repeated replication (BRR) variance estimation
vce(jackknife)	jackknife variance estimation
mse	use the MSE formula with vce(brr) or vce(jackknife)
singleunit(method)	strata with a single sampling unit; method may be missing, certainty, scaled, or centered
poststrata(varname)	variable identifying poststrata
postweight(varname)	poststratum population sizes

Das R-Package **survey** bietet den entsprechenden Befehl **svydesign**:

svydesign(ids, probs=NULL, strata = NULL, variables = NULL, fpc=NULL, data = NULL, nest = FALSE, check.strata = !nest, weights=NULL, pps=FALSE, variance=c("HT","YG"),...)

Die jeweilige Herangehensweise könnte wie folgt aussehen:

library(survey)

data(api)

stratified sample

dstrat=**svydesign**(id=~1,strata=~stype, weights=~pw, data=apistrat, fpc=~fpc)

one-stage cluster sample

dclus1=**svydesign**(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)

svy: regress

Modellschätzungen mit dem STATA Befehl **svy: regress** bzw. **regress** würden über die Syntax von **svy: regress** bzw**. regress** angegangen werden:

svy [vcetype] [, svy_options eform_option] : **regress**

webuse nhanes2f

svyset psuid [pweight=finalwgt], strata(stratid)

svy: regress zinc age c.age#c.age weight female black orace rural

Die üblichen *postestimation commands* bieten sich an:

estat (svy)	postestimation statistics for survey data
estimates	cataloging estimation results
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
margins	marginal means, predictive margins, marginal effects, and average marginal effects
nlcom	point estimates, SE, testing, and inference for nonlinear combinations of coefficients
predict	predictions, residuals, influence statistics, and other diagnostic measures
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
suest	seemingly unrelated estimation
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

Ein Beispiel für eine Regressionsschätzung in R mit dem Befehl **svyglm** (welcher auf glm aufbaut):

Syntax

svyglm(formula, design, subset=NULL, ...)

```
library(survey)
data(api)
dstratq=svydesign(id=~1,strata=~stype, weights=~pw, data=apistrat, fpc=~fpc)
summary(svyglm(api00~ell+meals+mobility, design=dstratq))
```

Es folgt die tabellarische Auflistung von einzelnen Befehlen / Befehlsstrukturen.

svyset Declare survey design for dataset, Single-stage syntax **svyset** [psu] [weight] [, design_options options] *One-stage clustered design with stratification* **webuse** stage5a **svyset** su1 [pweight=pw], strata(strata)	**svydesign** Specify a complex survey design **Syntax** **svydesign(ids, probs=NULL, strata = NULL, variables = NULL, fpc=NULL, data = NULL, nest = FALSE, check.strata = !nest, weights=NULL, pps=FALSE, variance=c("HT","YG"),...)** *one-stage cluster sample* library(survey) data(api) dclus1=svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
svy : command Prefix and command for statistical models for complex survey data. Syntax **svy** [vcetype] [, svy_options eform_option] **: command** *Jackknife variance estimation* **webuse** nhanes2 **svy** jackknife slope = _b[height] constant=_b[_cons]**: regress** weight height **Commands** Descriptive statistics **mean** Estimate means **proportion** Estimate proportions **ratio** Estimate ratios **total** Estimate totals	**svymean** Compute means for data from surveys Syntax **svymean(x, design, na.rm=FALSE,deff=FALSE,...)** *one-stage cluster sample* library(survey) data(api) dclus1=svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc) svymean(~interaction(stype, comp.imp), dclus1) svyquantile(~api00, dclus1, c(.25,.5,.75)) svytotal(~enroll, dclus1, deff=TRUE) svyratio(~api.stu, ~enroll, dclus1)
Linear regression models **cnsreg** Constrained linear regression **glm** Generalized linear models **intreg** Interval regression **nl** Nonlinear least-squares estimation **regress** Linear regression **tobit** Tobit regression **treatreg** Treatment-effects regression **truncreg** Truncated regression	**svyglm** Fit a generalised linear model to data from a complex survey design, with inverse-probability weighting and design-based standard errors **Syntax** **svyglm(formula, design, subset=NULL, ...)** library(survey) data(api) dstratq=svydesign(id=~1,strata=~stype, weights=~pw, data=apistrat, fpc=~fpc) summary(svyglm(api00~ell+meals+mobility, design=dstratq))

<table>
<tr><td>Survival-data regression models
stcox Cox proportional hazards model
streg Parametric survival models</td><td>svycoxph Fit a proportional hazards model to data from a complex survey design
Syntax
library(survey)
svycoxph(formula, design,subset=NULL, ...)</td></tr>
<tr><td>Binary-response regression models
biprobit Bivariate probit regression
cloglog Complementary log-log regression
hetprob Heteroskedastic probit regression
logistic Logistic regression, reporting odds ratios
logit Logistic regression, reporting coefficients
probit Probit regression
scobit Skewed logistic regression</td><td rowspan="2">svyolr Fit cumulative link models: proportional odds, probit, complementary log-log, and cauchit
Syntax
svyolr(formula, design, start, ...,
na.action = na.omit, method =
c("logistic", "probit", "cloglog", “cauchit"))
library(survey)
data(api)
dclus1=svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
dclus1=update(dclus1, mealcat=cut(meals, c(0,25,50,75,100)))
svyolr(mealcat~avg.ed+mobility+stype, design=dclus1)</td></tr>
<tr><td>Discrete-response regression models
clogit Conditional (fixed-effects) logistic regression
mlogit Multinomial (polytomous) logistic regression
mprobit Multinomial probit regression
ologit Ordered logistic regression
oprobit Ordered probit regression
slogit Stereotype logistic regression</td></tr>
<tr><td>Poisson regression models
gnbreg Generalized negative binomial regression
nbreg Negative binomial regression
poisson Poisson regression
zinb Zero-inflated negative binomial regression
zip Zero-inflated Poisson regression
ztnb Zero-truncated negative binomial regression
ztp Zero-truncated Poisson regression</td><td rowspan="3">poisson.survey-Option (zelig): Survey-weighted poisson regression
Syntax
z.out = zelig(Y ~ X1 + X2,
model = "poisson.survey", data = mydata)
library(Zelig)
library(survey)
data(api)
z.out1 = zelig(enroll ~ api99 + yr.rnd, model = "poisson.survey", weights = ~pw, data = apistrat)</td></tr>
<tr><td>Instrumental-variables regression models
ivprobit Probit model with endogenous regressors
ivregress Single-equation instrumental-variables regression
ivtobit Tobit model with endogenous regressors</td></tr>
<tr><td>Regression models with selection
heckman Heckman selection model
heckprob Probit model with sample selection</td></tr>
</table>

5. Verlaufsdatenanalyse

Hier wurden 3 Befehle, stset, stcox und streg stellvertretend für andere, ausgewählt.
Mit **stset**, vergleichbar **xtset**, werden entsprechende Verlaufsdatencharakteristika (Überlebenszeitvariable, Ausfall (failure), Gewichte etc.) von vornherein für den Datensatz definiert, und nicht erst beim Ausführen einzelner Befehle.
Die Syntaxen sind jeweils:
Single-record-per-subject survival data
stset timevar [if] [weight] [, single_options]
Multiple-record-per-subject survival data
stset timevar [if] [weight] [, id(idvar) failure(failvar[==numlist]) multiple_options]

In einem ähnlichen Setup-Kontext stehen die Befehle ctset, cttost, sttocc, sttoct, die nicht weiter ausgeführt wurden, genauso wie *postestimation commands* oder Befehle, die Datenmanipulation, Tabellierungen, Visualisierung, Diagnostiken etc. liefern (stphplot, stcurve, stgen, strate, sts graph, sts test).

Auf **stcox** und **streg** wird näher eingegangen. Cox-Regressionen schätzt man mit:
stcox [varlist] [if] [in] [, options]
Optionen (, options) zur Modellspezifikation, zur robusten Schätzung etc.können sein:

estimate	fit model without covariates
strata(varnames)	strata ID variables
shared(varname)	shared-frailty ID variable
offset(varname)	include varname in model with coefficient constrained to 1
breslow	use Breslow method to handle tied failures; the default
efron	use Efron method to handle tied failures
exactm	use exact marginal-likelihood method to handle tied failures
exactp	use exact partial-likelihood method to handle tied failures
tvc(varlist)	time-varying covariates
texp(exp)	multiplier for time-varying covariates
vce(vcetype)	vcetype may be robust, jackknife…
noadjust	do not use standard degree-of-freedom adjustment

Als *postestimation commands* sind möglich:

estat concordance	Harrell's C
stcurve	plot the survivor, hazard, and cumulative hazard functions
predict	predictions, residuals, influence statistics, and other diagnostic measures
predictnl	point estimates, SE, testing, and inference for generalized predictions

Nach Schätzung der Cox-Regression mit **stcox** bietet sich **stcurve** zum Plotten der Survival- oder Hazardfunktion an.

Parametrische Modelle kann man mit **streg** schätzen:
streg [varlist] [if] [in] [, options]
Auch hier bieten sich Modellspezifikationen an:

noconstant	suppress constant term
distribution(exponential)	exponential survival distribution
distribution(gompertz)	Gompertz survival distribution
distribution(loglogistic)	loglogistic survival distribution
distribution(llogistic)	synonym for distribution(loglogistic)
distribution(weibull)	Weibull survival distribution
distribution(lognormal)	lognormal survival distribution
distribution(gamma)	generalized gamma survival distribution
frailty(gamma)	gamma frailty distribution
frailty(invgaussian)	inverse-Gaussian distribution
time	use accelerated failure-time metric
strata(varname)	strata ID variable
offset(varname)	include varname in model with coefficient constrained to 1
shared(varname)	shared frailty ID variable
ancillary(varlist)	use varlist to model the first ancillary parameter
anc2(varlist)	use varlist to model the second ancillary parameter
constraints(constraints)	apply specified linear constraints
collinear	keep collinear variables

Beispiel:

Fit a Weibull survival model

webuse kva

stset failtime

streg load bearings, distribution(weibull)

Stset, stcox und streg können in R mit dem Package **survival** gut nachvollzogen werden.

Surv erzeugt ein entsprechendes Survivalobjekt, welches in Regressionen eingehen kann.

Syntax:

Surv(time, time2, event, type=c('right', 'left', 'interval', 'counting', 'interval2'), origin=0)

is.Surv(x)

Eine Anwendung mit parametrischer Regression (**survreg**) könnte wie folgt aussehen:

library(survival)

survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian**, dist='weibull'**, scale=1)

Für Cox-Regressionen steht **coxph** zur Verfügung. Die Syntax lautet:

coxph(formula, data=, weights, subset, na.action, init, control, method=c("efron","breslow","exact"), singular.ok=TRUE, robust=FALSE, model=FALSE, x=FALSE, y=TRUE, ...)

Plot und **predict** können ähnlich zu STATA Befehlen angewandt werden.

Der tabellarische Vergleich der 3 besprochenen Befehle folgt weiter unten.

stset Declare data to be survival-time data Syntax *Single-record-per-subject survival data* **stset** timevar [if] [weight] [, single_options] *Multiple-record-per-subject survival data* **stset** timevar [if] [weight] [, id(idvar) failure(failvar[==numlist]) multiple_options] **webuse** kva **stset** failtime	**Surv** Create a survival object, usually used as a response variable in a model formula **Syntax** **Surv(time, time2, event, type=c('right', 'left', 'interval', 'counting', 'interval2'), origin=0)** **is.Surv(x)** library(survival) with(lung, Surv(time, status)) Surv(heart$start, heart$stop, heart$event)
stcox Cox proportional hazards model via MLE Syntax **stcox** [varlist] [if] [in] [, options] *Fit Cox proportional hazards model* **webuse** kva **stset** failtime **stcox** load bearings	**coxph** Fit a Cox proportional hazards regression model **Syntax** **coxph(formula, data=, weights, subset, na.action, init, control, method=c("efron","breslow","exact"), singular.ok=TRUE, robust=FALSE, model=FALSE, x=FALSE, y=TRUE, ...)** *Create data set and fit a stratified model* library(survival) test1 = list(time=c(4,3,1,1,2,2,3), status=c(1,1,1,0,1,1,0), x=c(0,2,1,1,1,0,0), sex=c(0,0,0,0,1,1,1)) coxph(Surv(time, status) ~ x + strata(sex), test1)
streg Parametric survival models Syntax **streg** [varlist] [if] [in] [, options] *Fit a Weibull survival model* **webuse** kva **stset** failtime **streg** load bearings, distribution(weibull)	**survreg** Fit a parametric survival regression model **Syntax** **survreg(formula, data, weights, subset, na.action, dist="weibull", init=NULL, scale=0, control,parms=NULL,model=FALSE, x=FALSE, y=TRUE, robust=FALSE, score=FALSE, ...)** *Fit a weibull model* library(survival) survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian, dist='weibull', scale=1)

6. Zeitreihenanalyse

Aufgenommen wurden die STATA Befehle: tsset, arch, arima, dfuller, pperron, var, varbasic und varsoc.
Corrgram, cumsp, dfgls, fcast compute, newey, pergram, prais, rolling, tsfill, tssmooth, vargranger, varlmar, varnorm, varstable, varwle, vec, veclmar, vecrank, wntestb wurden nicht separat zitiert.

Mit dem Befehl **tsset** wird der vorliegende Datensatz als Zeitreihe definiert, erst danach kann man entsprechende Zeitreihenanalysen durchführen. Eine Verwendung wie **xtset** (für Paneldaten) ist möglich. Die Syntax lautet:
tsset timevar [, options]
tsset panelvar timevar [, options]
ID panelvar, timevar
webuse invest2
tsset company time

Mit **pperron** und **dfuller** kann man in STATA Einheitswurzeltests berechnen; **wntestb** bzw. **wntestq** führen Whitenoisetests durch; **tsline** plottet Zeitreihen, während man sich mit **corrgram** u.a. Autokorrelationen und PAC ausgeben lassen kann:
corrgram varname [if] [in] [, corrgram_options]
In R kann man mit **acf** oder **pacf** Autokorrelationen zeigen, mit **adf.test** und **pp.test** auf Einheitswurzel testen.
Mit **arch** können in STATA zahlreiche ARCH-Modelle geschätzt werden, die Syntax dazu lautet:
arch depvar [indepvars] [if] [in] [weight] [, options]
Ein ARCH-Modell mit 3 Lags würde man schätzen mit:
arch depvar, **arch(1/3)**
Ein GARCH(1,1)-Modell mit Kovariaten:
arch illinois indiana kentucky, **arch(1) garch(1)**
Ein EGARCH-Model mit ARMA-Termen:
arch D.ln_wpi, ar(1) ma(1 4) earch(1) egarch(1)

Als Optionen hat man u.a. zur Auswahl (nur Nennungen):

noconstant	suppress constant term
arch(numlist)	ARCH terms
garch(numlist)	GARCH terms
saarch(numlist)	simple asymmetric ARCH terms
tarch(numlist)	threshold ARCH terms
aarch(numlist)	asymmetric ARCH terms
narch(numlist)	nonlinear ARCH terms
narchk(numlist)	nonlinear ARCH terms with single shift
abarch(numlist)	absolute value ARCH terms
atarch(numlist)	absolute threshold ARCH terms
sdgarch(numlist)	lags of s_t
earch(numlist)	new terms in Nelson's EGARCH model
egarch(numlist)	lags of ln(s_t^2)
parch(numlist)	power ARCH terms
tparch(numlist)	threshold power ARCH terms
aparch(numlist)	asymmetric power ARCH terms
nparch(numlist)	nonlinear power ARCH terms
nparchk(numlist)	nonlinear power ARCH terms with single shift
pgarch(numlist)	power GARCH terms
arima(#p, #d, #q)	specify ARIMA(p,d,q) model for dependent variable
ar(numlist)	autoregressive terms of the structural model disturbance
ma(numlist)	moving-average terms of the structural model disturbances
het(varlist)	include varlist in the specification of the conditional variance
savespace	conserve memory during estimation

ARIMA-Modelle können in STATA geschätzt werden mit:

arima depvar [indepvars], ar(numlist) ma(numlist)

Für eine ARIMA(1,1,1)-Modellanpassung schreibt man:

arima wpi, arima(1,1,1)

Man passt ein multiplikatives SARIMA-Modell an, und unterdrückt den konstanten Term mit:

arima lnair, arima(0,1,1) sarima(0,1,1,12) noconstant

Zur Verfügung stehen u.a. folgende *postestimation commands*:

estat	AIC, BIC, VCE, and estimation sample summary
estimates	cataloging estimation results
lincom	point estimates, SE, testing, and inference for linear combinations of coefficients
lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects

Vektorautoregressive Modelle schätzt man mit:

var depvarlist [if] [in] [, options]

Modellschätzung mit 1.,2., und 3. Lag und *postestimation:*

webuse lutkepohl2

var dln_inv dln_inc dln_consump , lags(1/3)

varnorm

varsoc

Neben klassischen *postestimation commands* bieten sich im Kontext an:

fcast compute	obtain dynamic forecasts
fcast graph	graph dynamic forecasts obtained from fcast compute
irf	create and analyze IRFs and FEVDs
vargranger	Granger causality tests
varlmar	LM test for autocorrelation in residuals
varnorm	test for normally distributed residuals
varsoc	lag-order selection criteria
varstable	check stability condition of estimates
varwle	Wald lag-exclusion statistics

Die R-Packages dynlm, vars, tseries, urca, FitAR liefern zahlreiche Befehle zur Zeitreihenanalyse, von der Modellschätzung bis hin zur Diagnostik.

In R kann man z.B. **ts** zur Definition von Zeitreihen verwenden, Modellschätzungen über **arima** oder **garch** funktionieren aber auch mit der Eingabe numerischer Vektoren / Variablen; der Befehl hat in dem Sinne nicht die Bedeutung von **tsset** in STATA.

R schätzt ARIMA-Modelle mit dem Befehl:

arima(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0, 0), period = NA), xreg = NULL, include.mean = TRUE, transform.pars = TRUE, fixed = NULL, init = NULL, method = c("CSS-ML", "ML", "CSS"), n.cond, optim.method = "BFGS", optim.control = list(), kappa = 1e6)

Beispiel:

arima(USAccDeaths, order = c(0,1,1), seasonal = list(order=c(0,1,1)))

Vektorautoregressive Modelle passt man an mit:

VAR(y, p = 1, type = c("const", "trend", "both", "none"), season = NULL, exogen = NULL, lag.max = NULL, ic = c("AIC", "HQ", "SC", "FPE"))

Beispiel:

library(vars)

data(Canada)

VAR(Canada, p = 2, type = "trend")

Einschlägige Testverfahren / Plots in R sind unter anderem:

adf.test()	computes the Augmented Dickey-Fuller test (tseries)
Box.test()	computes the Box-Pierce / Ljung-Box test statistic for examining the null hypothesis of independence in a given time series (stats)
bds.test()	computes and prints the BDS test statistics (tseries)
bptest()	performs the Breusch-Pagan test for heteroskedasticity of residuals (lmtest)
dwtest()	performs the Durbin-Watson test for autocorrelation of residuals (lmtest)
jarque.bera.test()	Jarque-Bera test for normality (tseries)
kpss.test()	computes KPSS test for stationarity (tseries)
shapiro.test()	Shapiro-Wilk Normality Test (stats)

Es folgt die Auflistung der Einzelbefehle in STATA und R.

arch Autoregressive conditional heteroskedasticity (ARCH) family of estimators Syntax **arch** depvar [indepvars] [if] [in] [weight] [, options] *GARCH(1,1) model with covariates* **webuse** urates **arch** illinois indiana kentucky, arch(1) garch(1)	**garch** Fit a GARCH(p, q) time series model to the data by computing the maximum-likelihood estimates of the conditionally normal model **Syntax** **garch(x, order = c(1, 1), series = NULL, control = garch.control(...), ...)** library(tseries) data(EuStockMarkets) dax = diff(log(EuStockMarkets))["DAX"] dax.garch = garch(dax)
arima ARIMA, ARMAX, and other dynamic regression models *Basic syntax for an ARIMA(p,d,q) model* **arima** depvar, arima(#p,#d,#q) *Simple ARIMA model with differencing and autoregressive and moving-average components* **webuse** wpi1 **arima** wpi, arima(1,1,1)	**arima** Fit an ARIMA model **Syntax** **arima(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0, 0), period = NA), xreg = NULL, include.mean = TRUE, transform.pars = TRUE, fixed = NULL, init = NULL, method = c("CSS-ML", "ML", "CSS"), n.cond, optim.method = "BFGS", optim.control = list(), kappa = 1e6)** arima(USAccDeaths, order = c(0,1,1), seasonal = list(order=c(0,1,1)))
dfuller Augmented Dickey–Fuller unit-root test Syntax **dfuller** varname [if] [in] [, options] *DF, including 3 lagged differences and a trend term* **webuse** air2 **dfuller** air, lags(3) trend	**adf.test** Augmented Dickey-Fuller unit-root test **Syntax** **adf.test(x, alternative = c("stationary", "explosive"), k = trunc((length(x)-1)^(1/3)))** library(tseries) x = rnorm(1000) adf.test(x)
pperron Phillips–Perron unit-root test Syntax **pperron** varname [if] [in] [, options] *4 Newey-West lags, including a trend term in the associated regression* **webuse** air2 **pperron** air, lags(4) trend	**pp.test** Compute the Phillips-Perron test for the null hypothesis that x has a unit root **Syntax** **pp.test(x, alternative = c("stationary", "explosive"), type = c("Z(alpha)", "Z(t_alpha)"), lshort = TRUE)** library(tseries) x = rnorm(1000) pp.test(x)
tsset Declare data to be time-series data Syntax **tsset** timevar [, options] **tsset** panelvar timevar [, options] *ID panelvar, timevar* **webuse** invest2 **tsset** company time	**ts** Create time-series objects **Syntax** **ts(data = NA, start = 1, end = numeric(0), frequency = 1, deltat = 1, ts.eps = getOption("ts.eps"), class = , names =)** **as.ts(x, ...)** **is.ts(x)** ts(1:10, frequency = 4, start = c(1959, 2)) *# 2nd Quarter of 1959*

var Vector autoregressive models Syntax **var** depvarlist [if] [in] [, options] *(2 lags default)* **webuse** lutkepohl2 **var** dln_inv dln_inc dln_consump	**VAR** Estimation of a VAR by utilising OLS per equation **Syntax** **VAR(y, p = 1, type = c("const", "trend", "both", "none"), season = NULL, exogen = NULL, lag.max = NULL, ic = c("AIC", "HQ", "SC", "FPE"))** library(vars) data(Canada) VAR(Canada, p = 2, type = "trend")
varbasic Fit a simple VAR and graph IRFs or FEVDs Syntax **varbasic** depvarlist [if] [in] [, options] *Fit includes the first, second, and third lags in the model* **webuse** lutkepohl2 **varbasic** dln_inv dln_inc dln_consump, irf lags(1/3)	**irf** Compute the impulse response coefficients **Syntax** **irf(x, impulse = NULL, response = NULL, n.ahead = 10, ortho = TRUE, cumulative = FALSE, boot = TRUE, ci = 0.95, runs = 100, seed = NULL, ...)** library(vars) data(Canada) var.2c = VAR(Canada, p = 2, type = "const") irf(var.2c, impulse = "e", response = c("prod", "rw", "U"), boot =FALSE)
varsoc Obtain lag-order selection statistics (FPE, AIC etc) for VARs and VECMs *Preestimation syntax* **varsoc** depvarlist [if] [in] [, preestimation_options] **webuse** lutkepohl2 **varsoc** dln_inv dln_inc dln_consump	**VARselect** Infomation criteria and final prediction error for sequential increasing the lag order up to a VAR(p)-proccess **Syntax** **VARselect(y, lag.max = 10, type = c("const", "trend", "both", "none"), season = NULL, exogen = NULL)** library(vars) data(Canada) VARselect(Canada, lag.max = 5, type="const")